Extending the scope of eID technology

-Threats and opportunities in a commercial setting-

Vincent Naessens[i] and Bart De Decker[ii]
[i]*Katholieke Hogeschool Sint-Lieven, Research group IT, MSEC, Belgium*
[ii]*Katholieke Universiteit Leuven, Department of Computer Science, DistriNet, Belgium*

ABSTRACT
In 2002, Belgium has adopted an electronic identity card as one of the first countries in Europe. By the end of 2009, the roll-out of the eID card will be completed. This means that each Belgian citizen will possess an eID card. The card enables her to digitally prove her identity and to legally sign electronic documents. The Belgian eID card opens up new opportunities for the government, its citizens, service providers and application developers. The Belgian eID technology originally aimed at facilitating transactions between Belgian citizens and the government. Although many eID applications have been developed, the success of the Belgian eID technology has not been what was expected. Therefore, the Belgian government encourages developers to build commercial applications that use the eID card (for authentication or e-signatures). However, extending the scope of the Belgian eID technology from e-government to the commercial sector is no sinecure and not without risks.

Keywords: electronic identity technology, ubiquitous access, authentication, privacy, security

INTRODUCTION
Every Belgian citizen (older than 12) has an electronic identity card since 2009. The Belgian Electronic identity card (BeID) allows citizens to identify themselves, to authenticate and to sign electronic documents. The BeID technology originally aimed at facilitating transactions with the Belgian government. Using the BeID *authentication* and/or *e-signature* functionality, citizens can get access to personal information stored in governmental databases[1] (such as personal records at the National Registration Office), retrieve official documents[2] (such as proof of birth/life/residence/nationality), declare their taxes[3], report criminal offences, etc. The card also supports *identification* of the card holder to police forces and to authorized border control officials. Identification with the BeID avoids inconsistencies and results in more reliable governmental databases (e.g. no double entries for the same individual due to manual input errors). Moreover, the card technology impedes counterfeiting and hence, identity fraud. Orthogonal to the basic functionality of the card, user-friendliness and restriction of integration/deployment costs were crucial concerns. As mainly governmental applications[4] were targeted, privacy was less important[5]. These concerns had an impact on the design of the BeID card. For instance,

[1] See also Belgian National Registration website. http://www.ibz.rrn.fgov.be/.
[2] See also Mijn dossier. https://www.mijndossier.rrn.fgov.be/.
[3] Seel also Tax-on-web. http://www.taxonweb.be/.
[4] See also My Belgium e-government services. http://my.belgium.be/.
[5] See also Privacy Features of European eID Card Specifications. ENISA Position Paper. January 2009.

the user's address is stored in the chip and is not printed on the card. Hence, there is no need for issuing a new BeID when a user moves to another address (i.e. the address file can be updated). Furthermore, the card implements no access control mechanism to read out the stored picture, identity and address files. Hence, the police can easily retrieve the data while keeping the infrastructural costs minimal (a simple card reader suffices; no keys/certificates need to be installed or regularly updated). Another concern was the "simplicity to integrate the BeID in existing or new applications". Application developers should not be security experts, and hence, it should be easy to use the BeID as a means to authenticate to web services. Since TLS (SSL) is one of the paradigms for mutual authentication in web applications, it seemed appropriate to make the card TLS-compatible. The threat model and design decisions (driven by low cost, high usability and easy deployment) were reasonable in the initial setting (i.e. the e-government domain). However, these design decisions result in serious privacy and security risks when extending the BeID technology to other domains (e.g. the commercial sector) (Verhaeghe et al., 2008). Currently, many countries and regions are planning to introduce eID technology. Each of them will be confronted with similar design decisions. Testing and evaluating the technology within one domain and later extending it gradually to other domains seems a good strategy. This strategy is indeed reasonable to evaluate certain parameters (such as usability, performance and cost). Yet, changing the setting may also change the privacy and security risks.

This paper elaborates on those risks and presents (partial) solutions. The rest of this chapter is structured as follows. First, an overview of the BeID technology (the card, the middleware and existing BeID applications) is presented (see section 2). Second, the crucial barriers that delay and hinder the development of commercial eID applications are classified (see section 3) and some (ad hoc) solutions are presented. Next, more structural approaches are discussed to accelerate commercial eID applications with the current card. The requirements and solutions resulted from discussions with many SMEs and large companies in Flanders within the scope of a technology transfer project funded by the government. Reusable software extensions as well as a framework that integrates the crucial components for privacy-friendly eID applications are presented. It will be shown that those solutions may tackle some major weaknesses and will lead to *second generation BeID applications*. However, some security threats still remain and can only be solved by a different eID design. The current BeID is therefore compared with other approaches, namely a domain-specific approach and a service-specific approach (e.g. the German eID card). The alternatives are compared and evaluated on multiple parameters (infrastructural cost, performance, usability, security and privacy). This chapter ends with a general conclusion.

OVERVIEW OF THE BELGIAN EID TECHNOLOGY

The Belgian eID card (Stern, 2003) is a smart card that allows Belgian citizens to both visually and digitally prove their identity and to sign electronic documents. The eID card contains three files: (1) a digital picture of the card holder, (2) an identity file, which contains the basic identity information and a hash value of the picture file; and is signed by the National Registry, and (3) an address file which contains the card holder's current place of residence; it is also signed by the National Registry together with the identity file to guarantee the link between both files.

The card contains two public key pairs: one for authentication purposes, the other for e-signing.

The two private keys SK_{Auth} and SK_{Sig} are securely stored in a tamper-resistant part of the chip and can be activated with a PIN code. The corresponding public key (PK_{Auth} and PK_{Sig}) are each certified in a certificate further listing the card holder's name and her nation-wide unique identification number (i.e. the National Registration Number or NRN). The Belgian government offers a *middleware* package (Rommelaere, 2003) to facilitate interactions with the eID card. The middleware GUI allows end-users to read the files, to retrieve the certificates that are stored in the eID card and to change the PIN code. It also acts as an intermediary for all accesses to the eID card by other applications. When a document has to be signed, the middleware passes a hash of the document to the card. Similarly, a hash of the challenge is passed to the card when authentication is required. A middleware popup window appears whenever the user is required to enter his PIN code; depending on the card reader (with or without embedded PIN pad),

the PIN is read by the middleware and sent to the card or the card reader forwards the PIN directly to the card. Likewise, the middleware can verify the validity of certificates (using CRL[6] or OCSP). Note, however, that the *official middleware* is not essential: there are several commercial versions available and an application can always implement the middleware's functionality itself and directly interact with the card.

CLASSIFICATION OF EID BARRIERS

This section classifies the eID barriers according to five categories: *barriers with respect to authentication, those with respect to identification*, *infrastructural barriers*, *legal/ethical/social barriers* and *technological barriers*.

The translation of the requirements for the eID card into its design involved a few decisions that have a major impact on the security and privacy risks when the initial scope of the card is extended from the e-governmental sector to the commercial domain. Since the BeID card has to be "TLS-compatible", the authentication protocol should be *service-independent*. In this protocol, the service sends a challenge to the card, which responds with a signature on that challenge and the certificate chain, necessary for verifying the signature.
The second consequence of TLS-compliance is that the card must implement *single sign-on* (SSO), since the TLS-protocol may require re-authentication after a timeout. It means that the card holder has to input her PIN code once to activate the authentication key (SK_{Auth}); after that, the card will perform any number of authentications until the card is removed from the card reader or an explicit logoff command is sent to the card.
The second requirement, "easy deployment for police forces", is realized by the design decision that it should be possible to read out the card with a simple card reader. Hence, *no access control* (for reading) is implemented. Picture, identity file, address file and both certificates can be read without any restrictions (no PIN code or user confirmation is required). To avoid abuse by applications, the official middleware locks the card reader and prompts the card holder (in a popup window) for her consent when a read-request was received from an application. The locking of the card reader has been removed in future versions of the middleware, since it prevents that other smart cards are being used (e.g. bank cards).

Barriers with respect to authentication

The Belgian eID card has originally been designed to facilitate interactions between citizens and the government. Hence, the threat model is only relevant and reasonable for eID applications in this domain. Therefore, the usage of the eID card in other domains (commercial, financial[7], medical, . . .) will inevitably introduce new threats. Most of them are caused by design decisions based on "wrong" assumptions.

A prototypical attack in this category exploits the SSO "feature" of the card. Malicious applications can surreptitiously authenticate the card owner to other service providers. For instance, a dubious insurance company could build an applet that allows clients to access personal financial information (i.e. info about their insurance policies) from their home. The applet requires authentication with the eID card. However, once the PIN code has been given, the applet can secretly perform multiple authentications towards other services without further user interaction. Hence, it can retrieve personal information from governmental databases or other commercial sites to which the user has been registered via her eID card. For instance, the applet can retrieve loan and tax information or the user's profile at an Internet shop. In the original threat model, this is a less serious issue since governmental institutions already share a lot of data and,

[6] See also The Belgian certificate revocation list. http://status.eid.belgium.be/.
[7] See also The KeyTrade bank. http://www.keytrade.be/.

hence, these applications would not benefit from this exploit. However, even in this case, the card remains vulnerable to spyware. Such a program just waits for a program (usually a browser) to access a known e-government site which requires BeID authentication; the spyware can then abuse the BeID card to secretly login into other websites and steal or modify the card holder's personal information.

The SSO-attack will become more important since the number of BeID applications will inevitably increase. First, as the roll-out of the BeID card has been completed by the end of 2009, *commercial service providers* will be tempted to use the BeID card as a strong authentication means. For instance, the bank sector currently studies the feasibility to use the BeID card for accessing bank accounts and other financial information and for performing financial transactions (i.e. home banking). Second, the *government* also plans to develop new BeID applications. More specifically, personal medical information will become accessible using the BeID. Hence, malicious service providers can build even more detailed profiles of their customers using reliable and even more sensitive personal information.

One straightforward solution to tackle this weakness is to deactivate the authentication key (i.e. logoff) after each authentication, but this could result in users having to enter their PIN code several times when the authentication is based on TLS. Logging-off can be done by either *the middleware* or *the application* and will prevent external programs (such as spyware and viruses) to use the card for secret authentications but does not solve the problem of malicious applets since they can access the BeID card directly (without mediation of the middleware).
A better solution is to authenticate the card holder after the TLS-connection has been set up or to use alternative authentication protocols. However, mutual authentication over TLS is a well-known paradigm for application developers who are no security experts. Moreover, as many cards have already been distributed, it will be very difficult and costly to revise the SSO feature.

A second consequence of the "TLS-compliance" is that authentication is *server-independent*. It means that the signed challenge does not refer to the "intended" destination server. Hence, the card holder always remains in a state of uncertainty to which remote service she is authenticating. Even if the middleware intercepts every authentication request and prompts the card holder for her consent, the pop-up window can only show the name of the requesting program (usually the browser), but not the intended service. If the name of that service would be sent to the card together with the challenge, and both were signed by the card, the card holder could have absolute certainty when a card reader with separate display is used: that display could show the authentication request and the intended service; hence, the card holder could always abort unwanted authentications.
This "non-standard" authentication protocol is, however, not compatible with TLS, and was, therefore, not implemented. However, as the BeID card can be seen as a master key that opens doors to many databases and services, the card designers should at least have made it more difficult to secretly abuse the card.

A straightforward solution to counter this kind of abuse of the card does not exist. A minimal solution consists of compelling the service providers to maintain a history of authentications, and to present this history each time the card holder accesses the service. It would even be better to have the BeID card keep a record of the last *n* authentications, which could then be regularly inspected by the card holder via the middleware. Of course, the latter solution is only useful if the card is kept informed about the services to which it authenticates.

A last design decision (in the authentication context) that leads to new threats is related to the *authentication and signature certificates* that are stored in the card. Both certificates contain uniquely identifying information, such as the NRN and the name of the card holder. This decision carries the seed of a possible "Big Brother" situation. If *the same certificates are used across multiple domains*, a health care provider (such as a home care organization), a bank (which belongs to the financial sector) and a commercial service provider (such as a travel agency or a book shop) could exchange their users' profiles

and, hence, easily link these profiles. As a result, the travel agency can charge more expensive rates to wealthy clients; the bank can increase the life insurance's premium of people that need medical home care, etc. The exchange of information is especially important when mergers or takeovers happen.

A solution to this problem consists of keeping one (or two) certificates *per domain* on the BeID card (i.e. card holders authenticate (or sign) with a domain-specific certificate). Domain-specific unique identifiers should be used in these certificates instead of one global unique identifier. Current smartcards can already store a substantial number of certificates. Of course, this assumes that service providers first authenticate towards the card; their certificates then indicate to which domain they belong. The server certificates can be verified by the software/middleware on the client workstation or preferably on the BeID card (see also next section).

Barriers with respect to identification

A design decision with a major effect is the *lack of access control* to the different files and certificates that are stored in the BeID. There is limited access control: the files and certificates are only readable (by everyone) and cannot be modified except for the address file, which can only be updated by the National Registry after proper authentication towards the card. It was reasonable to require that authorities (e.g. police) can access the information in the eID card without restriction (no PIN code required), since an infringer or criminal could pretend to have forgotten his PIN code. Having the police authenticate towards the card is an alternative, but the maintenance cost would be high (to install private keys and corresponding certificates in all the devices used by the police force). The official middleware intercepts read-operations and requests (in a popup window) the card holder's consent. However, a malicious application can directly access the BeID card and thus circumvent this interception. Currently, the BeID is used in hotels to automatically fill in the hotel register and at container waste parks to verify that the card holder is a citizen of the local municipality. Since cautious users are well aware that these files might be read by malicious applications, they will be reluctant to use their eID card in dubious commercial applications (such as access to sauna or fitness centers). Since the card itself never requests the card holder's approval, it is impossible to prevent with 100% certainty this kind of threat. Hence, whenever the eID card is used, spyware might secretly read the content of the card and forward the information to criminal organizations, which may infer that the card holder is currently far from home and criminals may visit her residence.

A straightforward solution to this problem is to have the service provider authenticate towards the card before allowing access to these files. The service provider's certificate should restrict what can be retrieved from the card. There is of course the problem that the card cannot verify whether the certificate has been revoked or not, since the card does not have direct access to the Internet. However, (imperfect) solutions can be devised such as verification by the middleware, certificates with a short validity period or a modified OCSP-response which includes a challenge of the card.

A pervert side effect of having identity and address files that are signed by the National Registry is that such files have a higher "commercial" value, since the information is guaranteed by the government. It is to be expected that these files will be collected and sold for commercial purposes.

Infrastructural barriers

Lack of *flexible and advanced organizational procedures* also seriously hinders the extension of the scope of the BeID technology to new domains: e.g. the time-consuming recovery procedure when the BeID card is stolen, lost or defect is unacceptable for many commercial applications. Currently many companies study the feasibility of using the BeID to log in into their local intranet. To remedy this problem, the administration will in the future distribute "blank cards" which can be used for this purpose. These cards will only contain a key pair and certificate to allow for temporary authentication.

A second issue is the *card reader*. The official website lists simple (cheap) card readers with neither separate keypad nor display. With these card readers, users have to enter their PIN code in a popup window using the PC's keyboard. Hence, keyboard sniffers can capture the PIN and use it whenever the eID card is put in the card reader. More advanced card readers with separate keypad and display are necessary to reduce risks. Such readers can even solve the SSO problem by automatically deactivating the key pair after each authentication.

To increase the quality and trustworthiness of digital signatures, a *trusted authority* and *online Internet access* are required. More specifically, reliable timestamps are typically added to digital signatures to increase their trustworthiness. Without a timestamp –or with an incorrect one– a citizen can sign a contract and subsequently revoke his BeID certificates. Thereafter, he can claim that the signature was made after the certificates were revoked.

Finally, many web service providers authenticate to clients using TLS connections. However, a substantial subset of them does not possess *verifiable server certificates*. In some cases, server certificates are self-signed or expired. The government's PKI should also issue server certificates.

Legal, ethical and social barriers

Governmental BeID applications honour the *privacy legislation*. However, more restrictive privacy laws apply to commercial organizations. For instance, commercial organizations are not allowed to store the user's NRN. However, this nation-wide unique number is disclosed every time the BeID card is used for authentication or signature generation, since the NRN is embedded in every certificate and in the identity file. Hence, a company is not allowed to keep a record of the citizen's signing certificate, which is nevertheless necessary to be able to prove later that the citizen's signature is valid.

Ethical barriers may also hinder the use of the BeID technology. The BeID certificates contain the card holder's name and NRN. However, the NRN is a compound number: it consists of the date of birth and some extra digits which are odd for males and even for females. Hence, the receiver of the certificate can already derive the card holder's name, date of birth and gender. Even more information is kept in the identity and address files. In most cases, usage of the BeID card will reveal more personal information than is strictly required by the service. In the previous section, we already discussed the lack of access control; in this section, we present a solution to the unlimited disclosure of personal information.

Instead of keeping all the personal information in plaintext in the certificates and identity and address files, the information could be encrypted or hashed, and their plaintext values should only be revealed with the card holder's consent (assuming a secure card reader with separate keypad and display). The server certificates (used in the access control mechanism described in the previous section) could already restrict the list of accessible personal information. Hence, the card holder only has to further limit this list.

Technological barriers

Most SMEs that develop eID applications do not employ security experts. There is also only a limited support offered by the government. Examples of *primitive BeID applications* or *first generation BeID applications* (i.e. applications that use the BeID for every transaction or service consumption) are listed in the tutorial. The middleware allows developers to access the BeID through an intuitive API. Also, a reverse proxy is available to verify the validity of BeID certificates at the server side (using CRLs or OCSP) when TLS-based authentication is used. At the client side, the card holder needs only to install the middleware on his PC.

More advanced design and development skills are required to develop applications with stricter security, privacy and mobility requirements. *Advanced eID applications* or *second generation eID applications* use the BeID as bootstrap to obtain privacy-friendly credentials (such as pseudonym certificates or anonymous credentials). These credentials could be issued by a trusted identity provider when the user registers (with her BeID card) with the service provider. They can later be used with every transaction or

service consumption and since the BeID card is not necessary anymore, they can be transferred to mobile devices. Hence, the user does no longer need a card reader to authenticate. Although this approach increases the quality of BeID applications –in terms of privacy, security, mobility and usability– the design and development complexity of the applications may increase considerably. This is, however, a real impediment for many SMEs. Therefore, the eIDea research project (Naessens, 2009) has built (a) *reusable software components* and (b) a *framework with simple interfaces for application programmers*. The software components increase the usability and mobility properties of BeID applications and can bootstrap applications which require accountability/liability but allow for pseudonymity or even anonymity. The framework hides the complexity of cryptographic building blocks from application programmers. The rest of this paper focuses on concrete examples of technological support that has been developed during the eIDea project. The software support originates from real needs of SMEs. We also illustrate their feasibility with several examples.

REUSABLE SOFTWARE EXTENSIONS

The Belgian eID card allows users to authenticate and to digitally sign documents. The former is realized by signing the hash of a challenge sent by the server using the private key for authentication (i.e. $Sign_{eID}(Hash(challenge), Key_{Auth})$). A signature is generated by signing the hash of the document using the private key for signing (i.e. $Sign_{eID}(Hash(document), Key_{Sig})$). Note that the challenge and document are hashed by the middleware on the workstation. Only the hash value is sent to the card because of the limited processing and storage capabilities of smart cards. Often, security building blocks other than authentication or signing are required in many applications. This section describes how *symmetric keys* and *proxy certificates* can be generated using the eID technology (Lapon et al., 2009). Next, we demonstrate the applicability of both extensions and evaluate possible constraints.

Generation of secure symmetric keys

To generate a *symmetric key* based on the BeID, the user first generates an (possibly public) input string. The hash of that input string is signed with one of both private keys (SK_X) of the BeID card. The result is a *seed* to be used to generate a symmetric key *K*. *K* can then be used to encrypt data. The encrypted data and the input string can be stored locally or at a remote location. Table 1 shows the basic protocol.

Table 1. Generation of a secure symmetric key

U: *inputString* ← generateInput ( )
U: *seed* ← sign_{eID} (Hash(*inputString*), SK_X)
U: *K* ← generateSymmetricKey (*seed*)
U: *cipherText* ← encrypt (*plainText*, *K*)
U → S: send (*cipherText* || *inputString*)
S: store (*cipherText* || *inputString*)

This simple protocol opens up new opportunities for many application developers. The basic protocol is typically useful for securely storing credentials (tickets, passwords, private keys) or other confidential information on the user's workstation or at a remote location. Variants can be made based on the specific setting or the application in which the extension is used:

- SK_X can be either SK_{Auth} or SK_{Sig}. By using SK_{Auth} the card holder only has to enter her PIN code once and, hence, can then generate many seeds. This is particularly useful if each piece of data needs to be encrypted with a different secret key. SK_{Sig} is more secure since a PIN code is required for every generation of a seed.
- The *inputString* can either be a random value, a string that specifies the exact location where the *cipherText* will be stored, user info (such as the serial number of the BeID card) or a combination

thereof. If *inputString* specifies the exact location, then it is not necessary to append it to the stored *cipherText*. However, using a random value allows users to generate a new key, K^*, in case the previous key, K, is compromised.

- The protocol can be extended to hide the location (i.e. the *index*) at which the confidential data is stored. An attacker who succeeds to retrieve all (encrypted) records from the database cannot learn anything about who stored what in the database. The users generate the index as follows (see Table 2):

Table 2. Generation of a secure symmetric key and hidden index

… previous protocol (see Table 1) …
U: *index* ← Hash (*K*)
U → S: send (*cipherText*, *index*)
S: store *cipherText* at location *index*

- Confidential documents can also be shared among multiple users (i.e. a closed user group). One user, U_0, chooses a name for the group, which is also taken as the *inputString* and generates the secret key (see Table 1), K, with which the documents will be encrypted. Each other group member, U_i, then generates his own secret key, K_i, again using group name as *inputString* and calculates $X_i = K_i, \oplus K$. All X_is are published ($X_0 = 0$). Each group member U_i can access the files by regenerating his K_i, xoring it with his X_i and using the result as the decryption key.

This eID extension allows for ubiquitous access to shared/personal confidential information. The only prerequisite is a smart card reader. Of course, the card holder still has to trust the workstation as K is generated on the workstation. As SK_{Auth} and SK_{Sig} are card-specific, appropriate backup and recovery strategies must be worked out. K can be stored on a trusted device (such as an USB stick) and/or encrypted with a key that is derived from the user's PUK code (the card's unblocking code). Alternatively, escrow servers can be used. The merits and disadvantages of the alternatives are described by Bellare and Goldwasser (1997).

Generation of proxy certificates

The BeID card can be used to issue (i.e. sign) proxy certificates. A proxy certificate certifies the ownership of a public key. However, unlike ordinary certificates, proxy certificates are signed by another certificate of the owner and not by an external certification authority. It typically consists of a public key, the identity of the public key owner (i.e. the certificate holder) and has a limited validity period. It is signed with the private signature key of the BeID as shown in Table 3.

Table 3. Generating proxy certificates

U: (SK_U, PK_U) ← generateKeyPair( )
U: proxyCert ← generateProxyCertificate (PK_U, $Cert_{eID}$.Name, …, validity_period)
U: proxyCert.signature ← $sign_{eID}$ (proxyCert, SK_{Sig})

Proxy certificates allow individuals to delegate some of their rights to other devices and/or individuals. For instance, the proxy certificate, the corresponding private key SK_U and the BeID signing certificate can be stored on a mobile device. Hence, users can authenticate and/or sign documents using platforms where even no card reader is available. Similarly, an employer can delegate a proxy certificate (with limited *validity period* and *purpose*) and corresponding private key to an employee who is responsible for the company's tax declarations. Note that proxy certificates can also be used to securely distribute (public) encryption keys. To tackle major privacy and infrastructural concerns, BeID proxy certificates slightly deviate from the standards. These modification are discussed in detail in (Lapon et al., 2009).

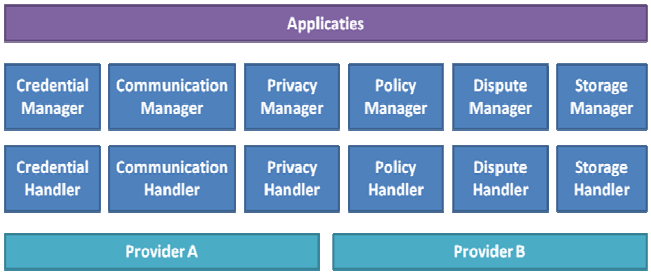
Proxy certificates are useful in many applications: two or more individuals can set up SSL/TLS connections without any intervention of a trusted Certification Authority (which is especially interesting in peer-to-peer applications); furthermore, they may solve management, usability and trust problems in secure mail protocols (such as S/MIME and PGP).

FRAMEWORK DEVELOPMENT

A framework[8] (Diaz et al., 2007) supports the development of Belgian eID applications. It offers application developers a generic interface to use more advanced building blocks while hiding their complexity. The framework consists of *managers* and *handlers*. A manager is responsible for one or more handlers of the same type. An application either directly instantiates and invokes methods of a handler or delegates these tasks to the corresponding manager. For instance, the application can select a particular type of connection and instantiate it directly (invoking methods of the *communicationHandler*). Or the application can call a communication manager. The latter selects the type of connection and manages the connection. The handlers in the framework are listed below (see also table 3):

- The *credential handler* defines a generic interface to perform actions using multiple types of credentials (like signing, authenticating, issuing, verifying …). Examples are the *BeIDHandler*, the *X509Handler*, the *pseudonymCertificateHandler*, the *IdemixHandler*[9](Camenisch et al., 2001) … The *BeIDHandler* allows users to authenticate and sign data with the Belgian eID while the *IdemixHandler* allows authenticating anonymously and signing data.
- The *communication handler* is responsible for sending and receiving messages between two entities. The handler provides a generic interface to set up different types of connections: (SSL over) TCP sockets, Bluetooth and NFC connections, anonymous communication channels[10] (like TOR and JAP), …
- The *privacy handler* manages local profiles which mirror the remote service provider's profiles. This way, the user can keep track of what personal information has been disclosed to these service providers.
- The *policy handler* analyses the service provider's privacy policy and tries to match it with the user's privacy preferences. The user is warned when personal data threatens to be disclosed which should remain hidden according to the privacy preferences.
- The *dispute handler* helps in collecting and providing evidence in case of a dispute between a user and service provider. Examples are signed liabilities, transcripts of authentication sessions, …
- The *storage handler* provides methods to store sensitive information such as tickets, personal data, credentials, evidence …) either locally or remotely.

Figure 1. The ADAPID framework



[8] See also The ADAPID project website. https://www.cosic.esat.kuleuven.be/adapid/.
[9] See also Idemix website. https://www.zurich.ibm.com/idemix.
[10] See also (Goldschlag, 1997)

A major goal of the framework is to accelerate the development of BeID applications of high quality. The applicability of the framework is already validated through the development of multiple applications[11]. The advantages offered by the framework are listed below:

- Each component has a uniform interface (to clients and servers) that supports multiple technologies. For instance, the *credentialHandler* interface supports different credential technologies, the *communicationHandler* interface supports multiple types of connections … Hence, application developers do not have to deal with the complexity of the building blocks and can easily replace instantiations with a minimal effort. For instance, replacing a *pseudonymHandler* by an *IdemixHandler* only requires a small modification in a configuration file.
- The framework is open to developers. Hence, developers can add new managers or handlers or provide an update of an existing implementation. Application programmers have to select a technology, a provider (which contains the actual implementations) and a version of an actual implementation.

OTHER APPROACHES AND FUTURE DIRECTIONS

It is by far not trivial to apply the Belgian eID technology across multiple domains. Using the card directly to authenticate to commercial services introduces serious security and threats (especially when the card is used at an untrusted platform). As discussed before, a feasible strategy for the Belgian eID consists of using the card only as a bootstrap to retrieve other (privacy-friendly) tokens. However, this strategy seriously complicates the design of new eID applications. The underlying shortcomings in the design of the BeID can be classified according to two categories:

a. *Lack of server authentication.* Servers do not have to authenticate to the card. The middleware could pop up a warning if servers do not authenticate successfully to the workstation. However, applications can circumvent the middleware. Moreover, many users ignore warnings caused by expired or invalid server certificates (i.e. they override the warnings).
b. *Coarse-grained access control.* Applications can read without any restrictions the information stored in the BeID (the picture, identity file, address file and certificates). They can also successfully authenticate the card holder to multiple servers as soon as the user enters his PIN code. In the authentication protocol, the user discloses the same set of personal information (independent of the particular service). Moreover, each time uniquely identifying personal information is released (e.g. the NRN). Hence, colluding service providers can easily build extensive profiles.

To address these privacy and security threats (which increase when the BeID card is used across multiple domains), alternative designs are more appropriate. The rest of this section evaluates and compares a *domain-specific* and a *service-specific* approach. Both strategies use partial identities. This means that citizens are known by a different pseudonym to each domain or service provider. Moreover, the personal attributes that are disclosed depend on the specific domain or service. For simplicity, this section only focuses on authentication (i.e. identification and digital signatures are not discussed). We also assume a card reader with separate key pad and display.

A *domain-specific approach* is presented by Verhaeghe et al. (2008). Instead of storing only one certificate (and corresponding private key) on the card, multiple certificates (and private keys) are stored. Each certificate is meant to be used in one particular domain (e.g. the financial domain, the governmental domain, the commercial domain, the medical domain …). Moreover, each certificate contains additional attrictutes that depend on the specific domain. For instance, the user's blood type is kept in his medical certificate whereas the NRN is kept in his governmental certificate. Each service provider (SP) also needs

[11] Lapon et al. (2008) have presented an ePetition system using the eID as a bootstrap; Verslype et al. (2008) have built a privacy-friendly ticketing system using the eID at the registration phase. Both applications are built upon the framework.

its own certificate issued by a domain-specific CA. The domain-specific CA is itself certified by a governmental CA (i.e. the root CA). The public key PK_{root} of the governmental CA is stored in each BeID card. Service providers first authenticate to the card. The certificate chain can be verified in the card by means of PK_{root}. Note that (1) the validity period of the server certificate and (2) its revocation status still need to be verified by the middleware on the user's workstation as (1) the smart card has no internal clock and (2) cannot set up a connection with a revocation server. If the service provider authenticates successfully to the card, the card authenticates the card holder towards the service provider using the right domain-specific certificate. Hence, only domain-specific identifiers and attributes are disclosed. Optionally, the attributes are not included in clear text into the certificate but are replaced by a randomized hash (i.e. *Hash(attribute_value_X || random_X)*). The clear text and random values (e.g. *attribute_value_X* and *random_X*) are also stored in the card. Hence, the card holder can select which attributes will be revealed to the service provider. Alternatively, the domain-specific CA can restrict the set of attributes that a service provider can ask for by specifying it in the service provider's certificate. A combined approach is possible. The user can then further reduce that set. This approach makes it harder to build profiles across multiple domains. One major disadvantage of this approach is that the number of domains (and attributes per domain) that can be supported depends on the storage capacity of the card. However, the storage capacity is not linear to the number of domains. Multiple optimizations are discussed in (Verhaeghe, 2009). For instance, the random values (used in the randomized hashes) can be derived from one secret master value; hence, only that master value needs to be stored in the card. Also, attribute values that are used in multiple domains, only have to be stored once.

In a *service-specific* approach, citizens are known by a different pseudonym to each service (provider). The previous approach with static pre-installed domain-specific certificates is neither flexible nor scalable enough to realize service-specific pseudonyms. A totally different approach will be used in the (contactless) German eID card. First, a mutually authenticated secure channel is set up between the card and the service provider. The card uses an asymmetric key pair that is common to all cards that are issued during a certain period. The server uses an access certificate that is issued by a public authority. Note that the citizen has to enter his PIN code before the card can verify the access certificate. After mutual authentication, the service provider can query the card. For instance, it can ask whether the citizen is older than 18. The access certificate contains access rights to data on the eID card. For instance, access to the card holder's address is restricted to service providers that need an address for their business. Note that the data disclosed by the eID card is not signed. Hence, this data will have no value to third parties (as it is not certified). Only the service provider can be sure about the authenticity of the data. Moreover, the card delivers a service-specific pseudonym based on a randomized UID of the chip and an identifier that is included in the server certificate of the service provider. The German design does not rely on a workstation to verify the validity and revocation status of server certificates. Instead, those cards have a notion of time. The validity of server certificates is limited (from a few days up to one month). The card stores the validity period *[T_start, T_end]* of the last seen server certificate. The card no longer accepts access certificates with an end date smaller than *T_start*. Although the German eID design offers a higher level of security and additional measures to protect the user's privacy compared to the Belgian eID, the deployment cost increases considerably:

a. Service providers have to renew their access certificates very frequently. Hence, the government CAs must be able to automate this procedure. However, not all services (e.g. cigarette machines) are online. This implies that an administrator has to interfere and visit these machines regularly.
b. Users are known by a different pseudonym to each service provider. This means that the governments must generate service-provider specific revocation lists. Those lists are not publicly available. Otherwise, revoked pseudonyms could be linked based on the revocation date. Hence, the government will have to interact with each service provider.

Table 3. Multiple approaches for designing smartcard based electronic identities

	User-specific identifier	Card-specific identifier
a. 1 global identifier	Belgian eID	
b. 1 nym per domain	Domain-specific approach	
c. 1 nym per service		German eID

Table 3 and table 4 compare the properties of the Belgian eID (BeID) to the domain-specific approach (DomID) and service-provider specific approach (ServID). Note that nyms in the German eID are card-specific. Also notice that the lifetime of server certificates is short in the German eID. So, many applications will not even bother to check revocation lists (see table 4).

Table 4. Verification of server certificates at the client side.

	At the card	At the workstation
a. Certificate chain	(DomID), (ServID)	(BeID)
b. Validity period	(ServID)	(BeID), (DomID)
c. Revocation status	--	(BeID), (DomID), *(ServID)*

The increased processing power and storage capacity of smart cards offer new opportunities. More advanced applications and cryptographic technologies can be stored on the card. There is a tendency towards moving more functionality from the workstation to the smartcard. This decreases the level of trust a user must have in the workstation. However, smart cards still have to deal with a number of drawbacks:

- Regular software updates are complicated as the card is only on-line when it is inserted in a card reader connected to a workstation that is connected to the Internet.
- Smartcards do not have a clock. Hence, the card has to rely on the workstation to check the validity of server certificates (or set up a secure connection with a trusted third party).
- A card reader with display or a workstation is required for user interaction. Again, trust is required in the card reader and/or workstation. Moreover, only primitive interactions are possible with many card readers.

Many countries therefore explore the possibility to use mobile devices for identity management. Examples can be found in (Enisa, 2008). Mobile devices have multiple benefits compared to smart cards: more storage space, processing power, communication options (GPRS, Bluetooth, NFC …) are available. Moreover, advanced user interaction is possible (i.e. identities can be managed, personal privacy preferences can be configured, connections can be blocked …). Hence, the user can have more control about his actions and the information that is released during transactions. Finally, a card reader and workstation are no longer required. This increases the mobility of identities. However, mobile devices can also be infected by malware (viruses, Trojan horses …). Users update the software on mobile devices less frequently than on their workstation.

CONCLUSION

Serious security and privacy risks impede the development of BeID applications (especially in the commercial domain). The design of the card was driven by a threat model that is reasonable for governmental applications. However, the threat model should be widened when extending the BeID technology to other domains (commercial, financial …). Hence, the current BeID card is a prototypical example of a technology that is to be utilized in domains for which it was not designed. The contribution of this chapter is threefold. First, the barriers that hinder the development of commercial BeID applications are classified. Second, structural solutions are also presented. Finally, the BeID is compared to alternative eID designs.

ACKNOWLEDGMENT

This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven, the IWT-SBO project ADAPID and the IWT-Tetra project eIDea.

REFERENCES

Andries P. eID (2003), *Middleware Architecture Document*. Zetes, 1.0 edition.
Bellare M. and Goldwasser S. (1997). Verifiable partial key escrow. In *Proceedings of the 4th ACM conference on Computer and communications security*, 78-91, New York, USA.
Brands S.(1999). A technical overview of digital credentials.
Camenisch J. and Lysyanskaya A. (2001). An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 93-118, Springer-Verlag, London, UK.
Chaum D. (1985). Security without identification: transaction systems to make big brother obsolete. *Commununications of the ACM*, 28(10):1030-1044.
Diaz C., De Decker B., Gevers S., Layouni M., Troncoso C., Van Es H., and Verslype K. (2007). *Advanced Applications for eID Cards in Flanders*, Deliverable 9: Framework I. Technical Report. Available at https://www.cosic.esat.kuleuven.be/adapid/documents.html
ENISA Position Paper: *Security Issues in the Context of Authentication Using Mobile Devices (Mobile eID)*, November 2008, ttp://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_mobile_eid.pdf
Federal Office for Information Security (BSI): Technical Guideline TR-03110, *Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC) and Password Authentication Connection Establishment (PACE), and Restricted Authentication, Version 2.0, (for national ID cards),* http://www.bsi.bund.de/english/publications/techguidelines/tr03110/TR-03110_v200.pdf
Federal Ministry of the Interior (BMI*): Einführung des elektronischen Personalausweises in Deutschland, Grobkonzept*, Version 2.0.
Goldschlag D.M., Syverson P. F. and Reed M. G. (1997). Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, IEEE Computer Society.
Lapon J., Verdegem B., Verhaeghe P., Naessens V., and De Decker B. (2009, June). Extending the Belgian eID technology with mobile security functionality. In *Proceedings of The First International ICST Conference on Security and Privacy in Mobile Information and Communication Systems*, LNICST, Springer, Turin, Italy.
Lapon J., Verslype K., Verhaeghe P., De Decker B., Naessens V. (2008). PetAnon: *A Fair and Privacy-Preserving Petition System*, In proceedings of FIDIS/IFIP Internet Security & Privacy Summer School.
Naessens V., Lapon J., Verdegem B., De Decker B., and Verhaeghe P (2009). *Developing Secure Applications with the Belgian eID technology*. Final Report. Available at https://www.msec.be/eidea/
Rommelaere J. (2003). Belgian *Electronic Identity Card Middleware Programmers Guide.* Zetes, 1.40.
Stern M. (2003). *Belgian Electronic Identity Card content.* Zetes, CSC, 2.2 edition.
Verhaeghe P., Lapon J., De Decker B., Naessens V. and Verslype K. Security and Privacy Improvements for the Belgian eID Technology. *In proceedings of the 24th IFIP International Information Security Conference*, Springer, May 18-20, 2009, Pafos, Cyprus.
Verhaeghe P., Lapon J., Naessens V. and De Decker B. (2008, June), *Security and Privacy Threats of the Belgian Electronic Identity Card and Middleware*. Paper presented at the EEMA European e-Identity Conference, Den Haag.

Verslype K., Nigusse G., De Decker B., Naessens V., Lapon J. and Verhaeghe P. (2008). A Privacy-Preserving Ticketing System, 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security. In *Lecture Notes in Computer Science series*, Springer.

KEY TERMS & DEFINITIONS

Authentication: establishing or confirming something (or someone) as *authentic*, that is, that claims made by or about the subject are true.
BeID technology: Belgian Electronic Identity Technology.
Certificate: an electronic document which uses a digital signature to bind together a public key with an identity.
Electronic signature: any legally recognized electronic means that indicates that a person adopts the contents of an electronic message.
Middleware: computer software that connects software components or applications.
Public key infrastructure (PKI): a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates.
Spyware: malware that is installed on computers that collects little bits information at a time about users without their knowledge.
Symmetric key: a key that is used for both encryption and decryption.